Categorical Phase Semantics for Linear Logic

Riley Shahar
Steve Zdancewic

Outline

Outline

- Categorical semantics, an introduction

Outline

- Categorical semantics, an introduction
- Linear logic and phase semantics

Outline

- Categorical semantics, an introduction
- Linear logic and phase semantics
- Our work: categorical phase semantics

Outline

- Categorical semantics, an introduction
- Linear logic and phase semantics
- Our work: categorical phase semantics
- Conclusion & future work

The Story

PLs *Logics* *Categories*

The Story

$$\mathit{PLs} \xrightarrow{\text{Curry-Howard}} \mathit{Logics} \qquad\qquad \mathit{Categories}$$

The Story

$$
\begin{tikzcd}
\textit{PLs} \arrow[r, "\text{Curry-Howard}"] & \textit{Logics} \arrow[r, bend left, "\text{Syntactic Category}"] & \textit{Categories}
\end{tikzcd}
$$

The Story

$$
\begin{tikzcd}
\textit{PLs} \arrow[r, "\text{Curry-Howard}"] & \textit{Logics} \arrow[rr, bend left, "\text{Syntactic Category}"] & & \textit{Categories} \arrow[ll, bend left, "\text{Internal Language}"]
\end{tikzcd}
$$

The Story

$$\begin{tikzcd}
\mathit{PLs} \arrow[r, leftrightarrow, "\text{Curry-Howard}"] & \mathit{Logics} \arrow[r, bend left, "\text{Syntactic Category}"] & \mathit{Categories} \arrow[l, bend left, "\text{Internal Language}"]
\end{tikzcd}$$

Signatures

Signatures

We have some *atomic propositions*

$$X, Y, Z, \ldots$$

Signatures

We have some *atomic propositions*

$$X, Y, Z, \ldots$$

some *propositional connectives*

$$\neg, \wedge, \vee, \rightarrow, T, F, \ldots.$$

Signatures

We have some *atomic propositions*

$$X, Y, Z, \ldots$$

some *propositional connectives*

$$\neg, \wedge, \vee, \rightarrow, T, F, \ldots.$$

These form a *signature* (*language*, *syntax*),

$$\mathcal{L} = \{X, \neg Y, (X \wedge Y) \rightarrow (T \vee X), \ldots\}.$$

Inference Rules

Inference Rules

We define some *inference rules* on our signature:

$$\frac{A \qquad B}{A \wedge B}\ \wedge\mathrm{I} \qquad\qquad \frac{A \qquad A \to B}{B}\ \to\!\mathrm{E}.$$

Inference Rules

We define some *inference rules* on our signature:

$$\frac{A \qquad B}{A \wedge B}\ \wedge\mathrm{I} \qquad\qquad \frac{A \qquad A \to B}{B}\ \to\mathrm{E}.$$

We can use these to write *proof trees:*

$$\frac{\dfrac{A \qquad A \to B}{B}\ \to E \qquad B \to C}{C}\ \to E.$$

Provability Semantics

The inference rules determine* a *entailment* relation $\vdash$ on $\mathcal{L}$:

$A \vdash B$ whenever B is provable from A.

Provability Semantics

The inference rules determine* a *entailment* relation $\vdash$ on $\mathcal{L}$:

$$A \vdash B \text{ whenever } B \text{ is provable from } A.$$

The *identity axiom* asserts

$$\frac{}{A \vdash A}\ \textsc{Id}.$$

Provability Semantics

The inference rules determine* a *entailment* relation $\vdash$ on $\mathcal{L}$:

$$A \vdash B \text{ whenever } B \text{ is provable from } A.$$

The *identity axiom* asserts

$$\frac{}{A \vdash A}\ \textsc{Id.}$$

The *cut rule* asserts

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}\ \textsc{Cut.}$$

Provability Semantics

The inference rules determine* a *entailment* relation $\vdash$ on $\mathcal{L}$:

$$A \vdash B \text{ whenever } B \text{ is provable from } A.$$

The *identity axiom* asserts

$$\frac{}{A \vdash A}\ \textsc{Id}.$$

The *cut rule* asserts

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}\ \textsc{Cut}.$$

Any logic with these rules has an associated preorder, its *provability semantics*.

Provability Semantics of (Classical) Conjunction

Provability Semantics of (Classical) Conjunction

What do we know about $A \wedge B$?

Provability Semantics of (Classical) Conjunction

What do we know about $A \wedge B$?

$$A \wedge B \vdash A \qquad A \wedge B \vdash B$$

Provability Semantics of (Classical) Conjunction

What do we know about $A \wedge B$?

$$A \wedge B \vdash A \qquad A \wedge B \vdash B$$

$$\frac{Z \vdash A \quad Z \vdash B}{Z \vdash A \wedge B}$$

Provability Semantics of (Classical) Conjunction

What do we know about $A \wedge B$?

$$A \wedge B \vdash A \qquad A \wedge B \vdash B$$

$$\frac{Z \vdash A \quad Z \vdash B}{Z \vdash A \wedge B}$$

So, conjunction is just the meet (infimum, glb).

Provability Semantics of (Classical) Conjunction

What do we know about $A \wedge B$?

$$A \wedge B \vdash A \qquad A \wedge B \vdash B$$

$$\frac{Z \vdash A \quad Z \vdash B}{Z \vdash A \wedge B}$$

So, conjunction is just the meet (infimum, glb).

You can keep going: you'll get a *Heyting algebra* or a *Boolean algebra*.

Proof Objects

We can talk about *proof objects*:

Proof Objects

We can talk about *proof objects*:

$$x : A \text{ means } x \text{ is a proof of } A.$$

Proof Objects

We can talk about *proof objects*:

$$x : A \text{ means } x \text{ is a proof of } A.$$

$$\Gamma \vdash x : A \text{ means } x \text{ is a proof of } A \text{ under some assumptions } \Gamma.$$

Proof Objects

We can talk about *proof objects*:

$$x : A \text{ means } x \text{ is a proof of } A.$$

$$\Gamma \vdash x : A \text{ means } x \text{ is a proof of } A \text{ under some assumptions } \Gamma.$$

Alternate notation:

$$\begin{array}{c} \\ \vdots\, x \\ A \end{array} \quad , \quad \begin{array}{c} \Gamma \\ \vdots\, x \\ A \end{array}$$

The Curry-Howard Isomorphism

$$\dfrac{\dfrac{\begin{matrix} \vdots\, x \\ A \end{matrix} \qquad \begin{matrix} \vdots\, y \\ B \end{matrix}}{A \wedge B}}{A}$$

The Curry-Howard Isomorphism

$$\dfrac{\dfrac{\begin{array}{c}\vdots\, x \\ A\end{array} \qquad \begin{array}{c}\vdots\, y \\ B\end{array}}{A \wedge B}}{A}$$

```
fn (x:A, y:B) -> A {
  p = (x, y);
  return fst p;
}
```

The Curry-Howard Isomorphism

$$\dfrac{\dfrac{\begin{matrix}\vdots\, x \\ A\end{matrix} \qquad \begin{matrix}\vdots\, y \\ B\end{matrix}}{A \wedge B}}{A}$$

```
fn (x:A, y:B) -> A {
  p = (x, y);
  return fst p;
}
```

These are both just x, and for the *same reason*.

The Curry-Howard Isomorphism

$$\dfrac{\dfrac{\begin{array}{c}\vdots\; x \\ A\end{array} \qquad \begin{array}{c}\vdots\; y \\ B\end{array}}{A \land B}}{A}$$

```
fn (x:A, y:B) -> A {
  p = (x, y);
  return fst p;
}
```

These are both just x, and for the *same reason*.

This is the *Curry-Howard isomorphism:*

Types are propositions; programs are proofs.

Proof Semantics

Proof Semantics

The *identity axiom* asserts

$$\frac{}{x : A \vdash x : A}\ \textsc{Id}.$$

Proof Semantics

The *identity axiom* asserts

$$\frac{}{x : A \vdash x : A}\ \textsc{Id.}$$

The *cut rule* asserts

$$\frac{\Gamma \vdash x : A \qquad A \vdash y : B}{\Gamma \vdash \mathrm{cut}(x, y) : B}\ \textsc{Cut.}$$

Proof Semantics

The *identity axiom* asserts

$$\frac{}{x : A \vdash x : A}\ \textsc{Id}.$$

The *cut rule* asserts

$$\frac{\Gamma \vdash x : A \qquad A \vdash y : B}{\Gamma \vdash \mathrm{cut}(x, y) : B}\ \textsc{Cut}.$$

Put another way,

$$\begin{tikzcd} A \arrow[loop above, "\textsc{Id}"] \end{tikzcd}$$

Proof Semantics

The *identity axiom* asserts

$$\frac{}{x : A \vdash x : A}\ \textsc{Id}.$$

Put another way,

$$\begin{tikzcd} A \arrow[loop above, "\textsc{Id}"] \end{tikzcd}$$

The *cut rule* asserts

$$\frac{\Gamma \vdash x : A \qquad A \vdash y : B}{\Gamma \vdash \mathrm{cut}(x, y) : B}\ \textsc{Cut}.$$

$$\Gamma \xrightarrow{x} A \xrightarrow{y} B.$$

Proof Semantics

The *identity axiom* asserts

$$\frac{}{x : A \vdash x : A}\ \textsc{Id}.$$

Put another way,

$$\begin{tikzcd} A \arrow[loop above, "\textsc{Id}"] \end{tikzcd}$$

The *cut rule* asserts

$$\frac{\Gamma \vdash x : A \qquad A \vdash y : B}{\Gamma \vdash \mathrm{cut}(x, y) : B}\ \textsc{Cut}.$$

$$\begin{tikzcd} \Gamma \arrow[r, "x"] \arrow[rr, bend right, dashed, "y \circ x"'] & A \arrow[r, "y"] & B. \end{tikzcd}$$

Proof Semantics

The *identity axiom* asserts

$$\frac{}{x : A \vdash x : A}\ \textsc{Id}.$$

The *cut rule* asserts

$$\frac{\Gamma \vdash x : A \qquad A \vdash y : B}{\Gamma \vdash \mathrm{cut}(x, y) : B}\ \textsc{Cut}.$$

Put another way,

$$\begin{tikzcd} A \arrow[loop above, "\textsc{Id}"] \end{tikzcd}$$

$$\begin{tikzcd} \Gamma \arrow[r, "x"] \arrow[rr, dashed, bend right, "y \circ x"'] & A \arrow[r, "y"] & B. \end{tikzcd}$$

These is a *category*! Any logic (or language) has its *syntactic category* as its *proof semantics*.

Categories

Categories

A B

C

A *category* consists of:

- Objects

Categories

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "h"'] & B \arrow[d, "g"] \\ & C \end{tikzcd}$$

A *category* consists of:

- Objects
- Morphisms

Categories

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "g \circ f = h"'] & B \arrow[d, "g"] \\ & C \end{tikzcd}$$

A *category* consists of:

- Objects
- Morphisms
- Composition

Categories

$$
\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "g\circ f = h"'] \arrow[loop above, "1_x"] & B \arrow[d, "g"] \arrow[loop above, "1_y"] \\
& C \arrow[loop right, "1_z"]
\end{tikzcd}
$$

A *category* consists of:

- Objects
- Morphisms
- Composition
- Identities

Proof Semantics of (Classical) Conjunction

Proof Semantics of (Classical) Conjunction

Again: what do we know about $A \wedge B$?

$$A$$

$$A \wedge B$$

$$B$$

Proof Semantics of (Classical) Conjunction

Again: what do we know about $A \wedge B$?

$$\begin{tikzcd}
 & A \\
A \wedge B \arrow[ur, "\wedge E_1"'] \arrow[dr, "\wedge E_2"] & \\
 & B
\end{tikzcd}$$

Proof Semantics of (Classical) Conjunction

Again: what do we know about $A \wedge B$?

$$
\begin{tikzcd}
 & & A \\
\Gamma \arrow[urr, bend left, "x"] \arrow[drr, bend right, "y"'] & A \wedge B \arrow[ur, "\wedge E_1"'] \arrow[dr, "\wedge E_2"] & \\
 & & B
\end{tikzcd}
$$

Proof Semantics of (Classical) Conjunction

Again: what do we know about $A \wedge B$?

$$
\begin{tikzcd}
 & & A \\
\Gamma \arrow[r, dashed] \arrow[urr, bend left, "x"] \arrow[drr, bend right, "y"'] & A \wedge B \arrow[ur, "\wedge \mathrm{E}_1"'] \arrow[dr, "\wedge \mathrm{E}_2"] & \\
 & & B.
\end{tikzcd}
$$

Proof Semantics of (Classical) Conjunction

Again: what do we know about $A \wedge B$?

$$\begin{tikzcd}
& & A \\
\Gamma \arrow[rru, bend left, "x"] \arrow[r, dashed] \arrow[rrd, bend right, "y"'] & A \wedge B \arrow[ru, "\wedge \mathrm{E}_1"'] \arrow[rd, "\wedge \mathrm{E}_2"] & \\
& & B.
\end{tikzcd}$$

This is the *categorical product.*

Proof Semantics of (Classical) Conjunction

Again: what do we know about $A \wedge B$?

$$
\begin{tikzcd}
 & & A \\
\Gamma \arrow[r, dashed] \arrow[urr, bend left, "x"] \arrow[drr, bend right, "y"'] & A \wedge B \arrow[ur, "\wedge \mathrm{E}_1"'] \arrow[dr, "\wedge \mathrm{E}_2"] & \\
 & & B.
\end{tikzcd}
$$

This is the *categorical product*.

Again, you can keep going: you'll get (something like) a *catesian closed category*.

Why can we do math with sets?

Why can we do math with sets?

We ground all of math in sets. But, sets are just some category, of sets and set-functions.

Why can we do math with sets?

We ground all of math in sets. But, sets are just some category, of sets and set-functions.

The categorical explanation is that sets form a *topos*.

Why can we do math with sets?

We ground all of math in sets. But, sets are just some category, of sets and set-functions.

The categorical explanation is that sets form a *topos*.

One important property is that *subsets are predicates*:

Why can we do math with sets?

We ground all of math in sets. But, sets are just some category, of sets and set-functions.

The categorical explanation is that sets form a *topos*.

One important property is that *subsets are predicates*:

$$A \subseteq X \to (x \mapsto x \in A)$$

Why can we do math with sets?

We ground all of math in sets. But, sets are just some category, of sets and set-functions.

The categorical explanation is that sets form a *topos*.

One important property is that *subsets are predicates*:

$$A \subseteq X \to (x \mapsto x \in A)$$
$$(P : X \to \{0, 1\}) \to \{x \in X : P(x) = 1\}.$$

Why can we do math with sets?

We ground all of math in sets. But, sets are just some category, of sets and set-functions.

The categorical explanation is that sets form a *topos*.

One important property is that *subsets are predicates*:

$$A \subseteq X \to (x \mapsto x \in A)$$
$$(P : X \to \{0, 1\}) \to \{x \in X : P(x) = 1\}.$$

Why can we do math with sets?

We ground all of math in sets. But, sets are just some category, of sets and set-functions.

The categorical explanation is that sets form a *topos*.

One important property is that *subsets are predicates*:

$$A \subseteq X \to (x \mapsto x \in A)$$
$$(P : X \to \{0,1\}) \to \{x \in X : P(x) = 1\}.$$

How can we frame this categorically?

Subobjects

Subobjects

Subsets have *inclusion functions*

$$\begin{aligned} i\colon A &\to X \\ a &\mapsto a \end{aligned}$$

which are *injective*.

Subobjects

Subsets have *inclusion functions*

$$\begin{aligned} i \colon A &\to X \\ a &\mapsto a \end{aligned}$$

which are *injective*.

Fix a class $\mathcal{M}$ of morphisms, which includes the identities and is closed under composition. A *subobject* A of an object X is a morphism

$$i : A \hookrightarrow X.$$

in $\mathcal{M}$.

Subobjects

Subsets have *inclusion functions*

$$\begin{aligned} i\colon A &\to X \\ a &\mapsto a \end{aligned}$$

which are *injective*.

Fix a class $\mathcal{M}$ of morphisms, which includes the identities and is closed under composition. A *subobject* A of an object X is a morphism

$$i : A \hookrightarrow X.$$

in $\mathcal{M}$.

If you pick $\mathcal{M}$ to be *monomorphisms*, you get the normal notion of subsets, subgroups, subspaces, etc.

Predicate Semantics

Predicate Semantics

Let $\mathsf{Sub}_{\mathcal{C}}(X)$ denote the subobjects of an object X of $\mathcal{C}$.

Predicate Semantics

Let $\mathsf{Sub}_{\mathcal{C}}(X)$ denote the subobjects of an object X of $\mathcal{C}$.

$\mathsf{Sub}_{\mathcal{C}}(X)$ is naturally ordered by compatible inclusion:

Predicate Semantics

Let $\mathsf{Sub}_{\mathcal{C}}(X)$ denote the subobjects of an object X of $\mathcal{C}$.

$\mathsf{Sub}_{\mathcal{C}}(X)$ is naturally ordered by compatible inclusion:

$$i \leq j \quad \text{iff}$$

Predicate Semantics

Let $\mathsf{Sub}_{\mathcal{C}}(X)$ denote the subobjects of an object X of $\mathcal{C}$.

$\mathsf{Sub}_{\mathcal{C}}(X)$ is naturally ordered by compatible inclusion:

$$i \leq j \quad \text{iff} \quad \begin{tikzcd} A \arrow[r, hook, "i"] & X \\ & B. \arrow[u, hook, "j"'] \end{tikzcd}$$

Predicate Semantics

Let $\mathsf{Sub}_{\mathcal{C}}(X)$ denote the subobjects of an object X of $\mathcal{C}$.

$\mathsf{Sub}_{\mathcal{C}}(X)$ is naturally ordered by compatible inclusion:

$$i \leq j \quad \text{iff} \quad \begin{tikzcd} A \arrow[r, hook, "i"] \arrow[dr, hook, dashed, "\exists!"'] & X \\ & B. \arrow[u, hook, "j"'] \end{tikzcd}$$

Predicate Semantics

Let $\mathsf{Sub}_{\mathcal{C}}(X)$ denote the subobjects of an object X of $\mathcal{C}$.

$\mathsf{Sub}_{\mathcal{C}}(X)$ is naturally ordered by compatible inclusion:

$$i \leq j \quad \text{iff} \quad \begin{tikzcd} A \arrow[r, hook, "i"] \arrow[dr, hook, dashed, "\exists!"'] & X \\ & B. \arrow[u, hook, "j"'] \end{tikzcd}$$

Predicate Semantics

Let $\mathsf{Sub}_{\mathcal{C}}(X)$ denote the subobjects of an object X of $\mathcal{C}$.

$\mathsf{Sub}_{\mathcal{C}}(X)$ is naturally ordered by compatible inclusion:

$$i \leq j \quad \text{iff} \quad \begin{tikzcd} A \arrow[r, hook, "i"] \arrow[dr, hook, dashed, "\exists!"'] & X \\ & B. \arrow[u, hook, "j"'] \end{tikzcd}$$

This is a preorder, since $\mathcal{M}$ is closed under composition and has identities. Hence, subobjects give provability semantics for a logic.

Predicate Semantics

Let $\mathsf{Sub}_{\mathcal{C}}(X)$ denote the subobjects of an object X of $\mathcal{C}$.

$\mathsf{Sub}_{\mathcal{C}}(X)$ is naturally ordered by compatible inclusion:

$$i \leq j \quad \text{iff} \quad \begin{tikzcd} A \arrow[r, hook, "i"] \arrow[dr, hook, dashed, "\exists!"'] & X \\ & B. \arrow[u, hook, "j"'] \end{tikzcd}$$

This is a preorder, since $\mathcal{M}$ is closed under composition and has identities. Hence, subobjects give provability semantics for a logic.

With sets, the power set is a Heyting (Boolean) algebra. Such categories are called *Heyting (Boolean) categories*. These categories form the *predicate semantics* of intuitionistic (clasical) logic.

Refinement Types

Refinement Types

Consider this program:

```
fn divide(a: int, b: nat) -> ratl {
  return a / b;
}
```

Refinement Types

Consider this program:

```
fn divide(a: int, b: nat) -> ratl {
  return a / b;
}
```

Is this safe?

Refinement Types

Consider this program:

```
fn divide(a: int, b: nat) -> ratl {
  return a / b;
}
```

Is this safe? No! Division by zero!

Refinement Types

Consider this program:

```
fn divide(a: int, b: nat) -> ratl {
  return a / b;
}
```

Is this safe? No! Division by zero!

We want to write

```
fn divide(a: int, b: (nat such that b > 0)) -> ratl {
  return a / b;
}
```

Refinement Types

Consider this program:

```
fn divide(a: int, b: nat) -> ratl {
  return a / b;
}
```

Is this safe? No! Division by zero!

We want to write

```
fn divide(a: int, b: (nat such that b > 0)) -> ratl {
  return a / b;
}
```

This is a predicate! So we probably want our type system to be a Heyting category. Category theory can tell us what we need to do to get that.

Linear Logic

Linear Logic

Due to Girard (1987).

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

```
fn x y => (x, y)
```
✓

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

```
fn x y => (x, y)      ✓
fn x => (x, x)        ✓
```

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

```
fn x y => (x, y)     ✓
fn x => (x, x)       ✓
fn f (x, y) => f x y ✓
```

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

```
fn x y => (x, y)     ✓
fn x => (x, x)       ✓
fn f (x, y) => f x y ✓
fn (x, y) => x       ✓
fn (x, y) => y       ✓
```

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

```
fn x y => (x, y)       ✓
fn x => (x, x)         ✓
fn f (x, y) => f x y   ✓
fn (x, y) => x         ✓
fn (x, y) => y         ✓
```

Linear Logic

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

`fn x y => (x, y)`	✓
`fn x => (x, x)`	✓
`fn f (x, y) => f x y`	✓
`fn (x, y) => x`	✓
`fn (x, y) => y`	✓

Linear Logic

fn x y $\multimap$ x $\otimes$ y	✓

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

`fn x y => (x, y)`	✓
`fn x => (x, x)`	✓
`fn f (x, y) => f x y`	✓
`fn (x, y) => x`	✓
`fn (x, y) => y`	✓

Linear Logic

fn x y $\multimap$ x $\otimes$ y	✓
fn x $\multimap$ x $\otimes$ x	✗

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

```
fn x y => (x, y)         ✓
fn x => (x, x)           ✓
fn f (x, y) => f x y     ✓
fn (x, y) => x           ✓
fn (x, y) => y           ✓
```

Linear Logic

```
fn x y ⊸ x ⊗ y           ✓
fn x ⊸ x ⊗ x             ✗
fn f (x ⊗ y) ⊸ f x y     ✓
```

Linear Logic

Due to Girard (1987).

Provides a *logic of limited resources*.

Classical Logic

`fn x y => (x, y)`	✓
`fn x => (x, x)`	✓
`fn f (x, y) => f x y`	✓
`fn (x, y) => x`	✓
`fn (x, y) => y`	✓

Linear Logic

fn x y $\multimap$ x $\otimes$ y	✓
fn x $\multimap$ x $\otimes$ x	✗
fn f (x $\otimes$ y) $\multimap$ f x y	✓
fn (x $\otimes$ y) $\multimap$ x	✗
fn (x $\otimes$ y) $\multimap$ y	✗

Phase Semantics

Phase Semantics

Also due to Girard.

Phase Semantics

Also due to Girard.

An early provability semantics for linear logic.

Phase Semantics

Also due to Girard.

An early provability semantics for linear logic.

Works with any *classical monoid* M.

Phase Semantics

Also due to Girard.

An early provability semantics for linear logic.

Works with any *classical monoid* M.

$$[\![A]\!] \subseteq M$$

Phase Semantics

Also due to Girard.

An early provability semantics for linear logic.

Works with any *classical monoid* M.

$$
\begin{aligned}
[\![A]\!] &\subseteq M \\
[\![A \otimes B]\!] &= \mathrm{cl}([\![A]\!][\![B]\!]) \\
&= \mathrm{cl}\{ab : a \in [\![A]\!], b \in [\![B]\!]\}
\end{aligned}
$$

Phase Semantics

Also due to Girard.

An early provability semantics for linear logic.

Works with any *classical monoid M*.

$$\begin{aligned}
[\![A]\!] &\subseteq M \\
[\![A \otimes B]\!] &= \mathrm{cl}([\![A]\!][\![B]\!]) \\
&= \mathrm{cl}\{ab : a \in [\![A]\!], b \in [\![B]\!]\} \\
[\![A \multimap B]\!] &= \{c \in M : \forall a \in [\![A]\!], ac \subseteq [\![B]\!]\}
\end{aligned}$$

Phase Semantics

Also due to Girard.

An early provability semantics for linear logic.

Works with any *classical monoid M*.

$$
\begin{aligned}
[\![A]\!] &\subseteq M \\
[\![A \otimes B]\!] &= \mathrm{cl}([\![A]\!][\![B]\!]) \\
&= \mathrm{cl}\{ab : a \in [\![A]\!], b \in [\![B]\!]\} \\
[\![A \multimap B]\!] &= \{c \in M : \forall a \in [\![A]\!], ac \subseteq [\![B]\!]\} \\
&= \bigcup \{C \subseteq M : [\![A]\!]C \subseteq [\![B]\!]\}
\end{aligned}
$$

Example Applications of Phase Semantics

Example Applications of Phase Semantics

Technical lemmas about linear logic [Girard 1987, Okada 1999].

Example Applications of Phase Semantics

Technical lemmas about linear logic [Girard 1987, Okada 1999].

Undecidability of boolean bunched implications [LG 2010].

Example Applications of Phase Semantics

Technical lemmas about linear logic [Girard 1987, Okada 1999].

Undecidability of boolean bunched implications [LG 2010].

Safety of concurrent constraint programs [FRS 2001, HB 2008].

Example Applications of Phase Semantics

Technical lemmas about linear logic [Girard 1987, Okada 1999].

Undecidability of boolean bunched implications [LG 2010].

Safety of concurrent constraint programs [FRS 2001, HB 2008].

Takeaway: phase semantics are *tunable*.

Quantale Semantics

Quantale Semantics

$$[\![A \otimes B]\!] = \mathrm{cl}([\![A]\!][\![B]\!]); \quad [\![A \multimap B]\!] = \bigcup\{C \subseteq M : [\![A]\!]C \subseteq [\![B]\!]\}$$

Quantale Semantics

$$[\![A \otimes B]\!] = \mathrm{cl}([\![A]\!][\![B]\!]); \quad [\![A \multimap B]\!] = \bigcup\{C \subseteq M : [\![A]\!]C \subseteq [\![B]\!]\}$$

Yetter (1990) observed that you don't need elements to do phase semantics.

Quantale Semantics

$$[\![A \otimes B]\!] = \mathrm{cl}([\![A]\!][\![B]\!]); \quad [\![A \multimap B]\!] = \bigcup\{C \subseteq M : [\![A]\!]C \subseteq [\![B]\!]\}$$

Yetter (1990) observed that you don't need elements to do phase semantics.

Quantales are preorders with a multiplication and joins (sups, lubs) which distribute over multiplication. Examples include the power set of any monoid.

Quantale Semantics

$$[\![A \otimes B]\!] = \mathrm{cl}([\![A]\!][\![B]\!]); \quad [\![A \multimap B]\!] = \bigcup\{C \subseteq M : [\![A]\!]C \subseteq [\![B]\!]\}$$

Yetter (1990) observed that you don't need elements to do phase semantics.

Quantales are preorders with a multiplication and joins (sups, lubs) which distribute over multiplication. Examples include the power set of any monoid.

Quantale semantics are the provability semantics of linear logic.

Categorical Semantics of Linear Logic

Linear logic has categorical semantics in *closed symmetric monoidal categories*.

Linear logic has categorical semantics in *closed symmetric monoidal categories*.

Intuitionistic Logic

Linear logic has categorical semantics in *closed symmetric monoidal categories*.

Intuitionistic Logic

Semantics	Structure
Provability	Heyting Algebras

Linear logic has categorical semantics in *closed symmetric monoidal categories*.

Intuitionistic Logic

Semantics	Structure
Provability	Heyting Algebras
Proof	CCCs

Linear logic has categorical semantics in *closed symmetric monoidal categories*.

Intuitionistic Logic

Semantics	Structure
Provability	Heyting Algebras
Proof	CCCs
Predicate	Heyting Categories

Categorical Semantics of Linear Logic

Linear logic has categorical semantics in *closed symmetric monoidal categories*.

Intuitionistic Logic

Semantics	Structure
Provability	Heyting Algebras
Proof	CCCs
Predicate	Heyting Categories

Linear Logic

Categorical Semantics of Linear Logic

Linear logic has categorical semantics in *closed symmetric monoidal categories*.

Intuitionistic Logic

Semantics	Structure
Provability	Heyting Algebras
Proof	CCCs
Predicate	Heyting Categories

Linear Logic

Semantics	Structure
Provability	Quantales

Categorical Semantics of Linear Logic

Linear logic has categorical semantics in *closed symmetric monoidal categories*.

Intuitionistic Logic

Semantics	Structure
Provability	Heyting Algebras
Proof	CCCs
Predicate	Heyting Categories

Linear Logic

Semantics	Structure
Provability	Quantales
Proof	CSMCs

Categorical Semantics of Linear Logic

Linear logic has categorical semantics in *closed symmetric monoidal categories*.

Intuitionistic Logic

Semantics	Structure
Provability	Heyting Algebras
Proof	CCCs
Predicate	Heyting Categories

Linear Logic

Semantics	Structure
Provability	Quantales
Proof	CSMCs
Predicate	??

Our Contribution

We construct a predicate semantics of linear logic, capturing phase semantics as a special case.

Our Contribution

We construct a predicate semantics of linear logic, capturing phase semantics as a special case.

- We need a multiplicative structure on subobjects. We can get this via *monoid objects*.

Our Contribution

We construct a predicate semantics of linear logic, capturing phase semantics as a special case.

- We need a multiplicative structure on subobjects. We can get this via *monoid objects*.
- We need joins on subobjects. We can get this via *coproducts*.

Our Contribution

We construct a predicate semantics of linear logic, capturing phase semantics as a special case.

- We need a multiplicative structure on subobjects. We can get this via *monoid objects*.
- We need joins on subobjects. We can get this via *coproducts*.

To make subobjects closed under these operations, we need to take *images*.

A Worked Example

A Worked Example

Define

$$\texttt{List}X := \{\text{lists with elements in } X\}$$

A Worked Example

Define

$$\begin{aligned} \texttt{List}X &:= \{\text{lists with elements in } X\} \\ \texttt{Maybe}X &:= X \sqcup \{\bot\} \end{aligned}$$

A Worked Example

Define

$$\begin{aligned}
\mathtt{List}X &:= \{\text{lists with elements in } X\} \\
\mathtt{Maybe}X &:= X \sqcup \{\bot\} \\
\mathtt{List}_{2\mathbb{N}}X &:= \{l \in \mathtt{List}X : 2 \mid \mathtt{len}(l)\}
\end{aligned}$$

A Worked Example

Define

$$\mathtt{List}X := \{\text{lists with elements in } X\}$$
$$\mathtt{Maybe}X := X \sqcup \{\bot\}$$
$$\mathtt{List}_{2\mathbb{N}}X := \{l \in \mathtt{List}X : 2 \mid \mathtt{len}(l)\}$$

Let's compute $\mathtt{List}_{2\mathbb{N}} \otimes \mathtt{Maybe}$.

A Worked Example

Define

$$\mathtt{List}X := \{\text{lists with elements in } X\}$$

$$\mathtt{Maybe}X := X \sqcup \{\perp\}$$

$$\mathtt{List}_{2\mathbb{N}}X := \{l \in \mathtt{List}X : 2 \mid \mathtt{len}(l)\}$$

Let's compute $\mathtt{List}_{2\mathbb{N}} \otimes \mathtt{Maybe}$. Let $X = \{1, 2\}$.

A Worked Example

Define

$$\texttt{List}X := \{\text{lists with elements in } X\}$$
$$\texttt{Maybe}X := X \sqcup \{\perp\}$$
$$\texttt{List}_{2\mathbb{N}}X := \{l \in \texttt{List}X : 2 \mid \texttt{len}(l)\}$$

Let's compute $\texttt{List}_{2\mathbb{N}} \otimes \texttt{Maybe}$. Let $X = \{1, 2\}$. Compute $\texttt{List}_{2\mathbb{N}} \circ \texttt{Maybe}X$

A Worked Example

Define

$$\texttt{List}X := \{\text{lists with elements in } X\}$$

$$\texttt{Maybe}X := X \sqcup \{\perp\}$$

$$\texttt{List}_{2\mathbb{N}}X := \{l \in \texttt{List}X : 2 \mid \texttt{len}(l)\}$$

Let's compute $\texttt{List}_{2\mathbb{N}} \otimes \texttt{Maybe}$. Let $X = \{1, 2\}$. Compute $\texttt{List}_{2\mathbb{N}} \circ \texttt{Maybe}X$

$$= \{ \quad [], \quad [1, 1], \qquad [1, \perp], \quad [\perp, 2], \quad [\perp, \perp], \quad \ldots \}$$

A Worked Example

Define

$$\mathtt{List}X := \{\text{lists with elements in } X\}$$
$$\mathtt{Maybe}X := X \sqcup \{\bot\}$$
$$\mathtt{List}_{2\mathbb{N}}X := \{l \in \mathtt{List}X : 2 \mid \mathtt{len}(l)\}$$

Let's compute $\mathtt{List}_{2\mathbb{N}} \otimes \mathtt{Maybe}$. Let $X = \{1, 2\}$. Compute $\mathtt{List}_{2\mathbb{N}} \circ \mathtt{Maybe}X$

$$= \{ \quad [], \quad [1,1], \qquad [1,\bot], \quad [\bot,2], \quad [\bot,\bot], \quad \ldots\}$$
$$\simeq \{ \quad [], \quad [[1],[1]], \quad [[1],[]], \quad [[],[2]], \quad [[],[]], \quad \ldots\} \quad \text{by including}$$

A Worked Example

Define

$$\mathtt{List}X := \{\text{lists with elements in } X\}$$

$$\mathtt{Maybe}X := X \sqcup \{\bot\}$$

$$\mathtt{List}_{2\mathbb{N}}X := \{l \in \mathtt{List}X : 2 \mid \mathtt{len}(l)\}$$

Let's compute $\mathtt{List}_{2\mathbb{N}} \otimes \mathtt{Maybe}$. Let $X = \{1, 2\}$. Compute $\mathtt{List}_{2\mathbb{N}} \circ \mathtt{Maybe}X$

$$\begin{array}{llllllll} = \{ & [], & [1,1], & [1,\bot], & [\bot,2], & [\bot,\bot], & \dots\} & \\ \simeq \{ & [], & [[1],[1]], & [[1],[]], & [[],[2]], & [[],[]], & \dots\} & \text{by including} \\ \rightsquigarrow \{ & [], & [1,1], & [1], & [2], & [], & \dots\} & \text{by concating} \end{array}$$

A Worked Example

Define

$$\mathtt{List}X := \{\text{lists with elements in } X\}$$

$$\mathtt{Maybe}X := X \sqcup \{\bot\}$$

$$\mathtt{List}_{2\mathbb{N}}X := \{l \in \mathtt{List}X : 2 \mid \mathtt{len}(l)\}$$

Let's compute $\mathtt{List}_{2\mathbb{N}} \otimes \mathtt{Maybe}$. Let $X = \{1, 2\}$. Compute $\mathtt{List}_{2\mathbb{N}} \circ \mathtt{Maybe}X$

$$\begin{aligned}
&= \{ \quad [], \quad [1,1], \quad [1,\bot], \quad [\bot,2], \quad [\bot,\bot], \quad \dots \} \\
&\simeq \{ \quad [], \quad [[1],[1]], \quad [[1],[]], \quad [[],[2]], \quad [[],[]], \quad \dots \} \quad \text{by including} \\
&\rightsquigarrow \{ \quad [], \quad [1,1], \quad [1], \quad [2], \quad [], \quad \dots \} \quad \text{by concating} \\
&\rightsquigarrow \{ \quad [], \quad [1,1], \quad [1], \quad [2], \quad \dots \} \quad \text{by quotienting.}
\end{aligned}$$

A Worked Example

Define

$$\begin{aligned}
\texttt{List}X &:= \{\text{lists with elements in } X\} \\
\texttt{Maybe}X &:= X \sqcup \{\bot\} \\
\texttt{List}_{2\mathbb{N}}X &:= \{l \in \texttt{List}X : 2 \mid \texttt{len}(l)\}
\end{aligned}$$

Let's compute $\texttt{List}_{2\mathbb{N}} \otimes \texttt{Maybe}$. Let $X = \{1, 2\}$. Compute $\texttt{List}_{2\mathbb{N}} \circ \texttt{Maybe}X$

$$\begin{array}{llllllll}
= \{ & [], & [1,1], & [1,\bot], & [\bot,2], & [\bot,\bot], & \ldots\} & \\
\simeq \{ & [], & [[1],[1]], & [[1],[]], & [[],[2]], & [[],[]], & \ldots\} & \text{by including} \\
\rightsquigarrow \{ & [], & [1,1], & [1], & [2], & [], & \ldots\} & \text{by concating} \\
\rightsquigarrow \{ & [], & [1,1], & [1], & [2], & & \ldots\} & \text{by quotienting.}
\end{array}$$

Generalizing, $\texttt{List}_{2\mathbb{N}} \otimes \texttt{Maybe} = \texttt{List}$.

Future Work

Future Work

Study these semantics in specific settings

Future Work

Study these semantics in specific settings

- Giry monad

Future Work

Study these semantics in specific settings

- Giry monad
- Linear refinement types

Future Work

Study these semantics in specific settings

- Giry monad
- Linear refinement types

Study completeness and decidability via these semantics

Future Work

Study these semantics in specific settings

- Giry monad
- Linear refinement types

Study completeness and decidability via these semantics

Completely characterize predicate semantics for linear logic

Summary

Summary

Our construction turns subobject posets of monoid objects in certain categories into models of linear logic: a form of *predicate semantics for linear logic*.

Summary

Our construction turns subobject posets of monoid objects in certain categories into models of linear logic: a form of *predicate semantics for linear logic*.

Special cases give a linearization of monads, as in the previous example; Girard's phase semantics; and nondeterministic phase semantics.

Summary

Our construction turns subobject posets of monoid objects in certain categories into models of linear logic: a form of *predicate semantics for linear logic*.

Special cases give a linearization of monads, as in the previous example; Girard's phase semantics; and nondeterministic phase semantics.

Concretely, these semantics give a useful tool for analyzing type systems which depend on linear predicates for proof.

Summary

Our construction turns subobject posets of monoid objects in certain categories into models of linear logic: a form of *predicate semantics for linear logic*.

Special cases give a linearization of monads, as in the previous example; Girard's phase semantics; and nondeterministic phase semantics.

Concretely, these semantics give a useful tool for analyzing type systems which depend on linear predicates for proof.

Thanks! :)